Penerapan Algoritma *Viral System*pada *Single-Machine Total Weighted Tardiness Problem*

Umar Affandi, Budi Santosa
Jurusan Teknik Industri, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya
Kampus ITS Sukolilo Surabaya 60111
Email: budi_s@ie.its.ac.id

***Abstrak*--*Single Machine Total Weighted Tardiness Problem* (SMTWTP) merupakan permasalahan klasik kombinatorial yang dikenal np-*hard*. Pada penelitian ini, suatu algoritma yang relatif baru yang terinspirasi dari sistem replikasi virus yang disebut sebagai *Viral Systems* digunakan untuk menyelesaikan permasalahan tersebut. Algoritma dengan proses pencarian terdiri dari *Neighborhood* dan mutasi tersebut memiliki delapan parameter. Penelitian ini menerapkan algoritma *Viral Systems* pada SMTWTP. Pengujian dilakukan untuk menganalisa parameter dan performansi algoritma dalam penyelesaian permasalahan. Hasil eksperimen menunjukkan bahwa setiap parameter memberikan pengaruh masing-masing terhadap algoritma dalam sisi hasil dan waktu komputasi. Eksperimen terhadap set data 40 pekerjaan, 50 pekerjaan, dan 100 pekerjaan menampilkan hasil bahwa algoritma dapat menyelesaikan 235 solusi optimal dari 275 permasalahan.**

Kata kunci: *Viral Systems, Single-Machine Total Weighted Tardiness Problem*

I. PENDAHULUAN

PENJADWALAN dan penyusunan merupakan suatu bentuk pengambilan keputusan yang memiliki peranan penting dalam industri manufaktur maupun jasa. Dalam lingkungan kompetitif yang telah ada, pengurutan dan penjadwalan telah menjadi suatu hal yang penting untuk mempertahankan pangsa pasar. Perusahaan harus menentukan waktu pengiriman yang tepat berdasarkan perjanjian yang telah dilakukan terhadap konsumen. Kegagalan pelaksanaan hal tersebut dapat menyebabkan hilangnya loyalitas. Oleh karena itu, mereka perlu menjadwalkan aktifitas berdasarkan sumber daya yang tersedia dengan cara yang efisien [1].

Machine scheduling problem (MSP) merupakan salah satu model penjadwalan klasik. Permasalahan ini dapat dijumpai dalam berbagai permasalahan seperti *flexible manufacturing system* (FMS), perencanaan produksi, dan *airline industry*[2]. *Total weighted tardiness scheduling problem* merupakan bagian MSP yang tidak hanya merupakan permasalahan *NP-hard* [3], tapi juga cukup sulit jika dilihat dari sisi praktis. Potts dan Van Wassenhove [4] telah menemukan masalah ketika menyelesaikan permasalahan dengan 50 job untuk optimalisasi dengan menggunakan *state of the art branch and bound algorithm* [5].

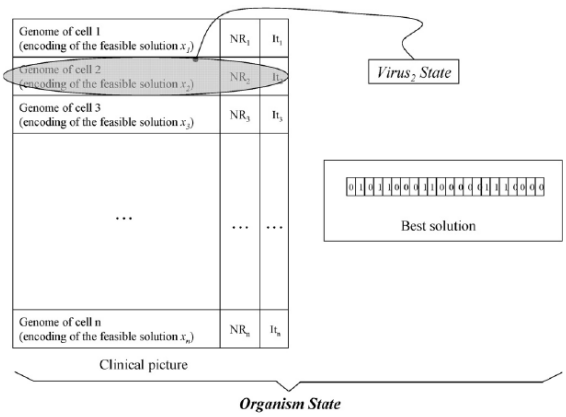
Sehubungan dengan sulitnya permasalahan ini, maka metaheuristik direkomendasikan untuk mencari solusi yang baik dengan alasan waktu komputasi. Metaheuristik telah banyak berhasil diterapkan untuk masalah ini termasuk GA, TS, dan ACO [6]. Pada penelitian ini, *Viral Systems* diajukan untuk mengatasi permasalahan SMTWTP.

Viral system merupakan algoritma optimasi berbasis metaheuristik yang diperkenalkan oleh Cortes, dkk. [7]. Algoritma ini merupakan metode yang relatif baru dan terinspirasi dari analogi sistem kinerja virus. Virus senantiasa bereproduksi dengan cara melakukan infeksi terhadap sel organisme dan organisme pun berusaha memberikan perlawanan berupa antobodi.

II. PENJELASAN ALGORITMA *VIRAL SYSTEMS*

Suatu makhluk transisi antara makhluk hidup dan benda mati bernama virus memiliki keunikan dalam hal reproduksi. Virus menggunakan sel organisme sebagai wadah untuk melakukan replikasi. Ketika virus melakukan injeksi DNA ke dalam tubuh sel inang, terdapat dua jenis daur replikasinya yaitu daur litik dan lisogenik. Daur litik memungkinkan virus untuk melakukan replikasi di dalam sel inang. Pada jumlah replikasi tertentu virus akan memecahkan dinding sel inang dan keluar. Sedangkan daur litik memungkinkan virus tetap berada dalam sel inang dalam keadaan nonaktif hingga ada sesuatu yang mengaktifkannya. Cepat atau lambatnya sel organisme yang telah terinjeksi tadi pecah atau termutasi, itu tergantung dari tingkat kesehatan dari sel itu sendiri.

Pada sistem virus, virus senantiasa mencari dan menginfeksi sel yang rentan. Setiap sel pun berusaha untuk mengeluarkan antibodi untuk melawan adanya infeksi virus. Virus yang gagal dalam mengeluarkan antibodi akan dikelompokkan kedalam *clinical picture*. VS mendefinisikan *clinical picture* dari populasi yang terinfeksi sebagai deskripsi dari semua sel yang terinfeksi oleh virus. Dalam hal komputasi, *Clinical Picture*berisi solusi yang sedang dieksplorasi (genom sel yang terinfeksi, dalam hal biologis) dan jumlah inti-capsids yang direplikasi yang berupa NR (untuk replikasi litik) atau IT (untuk replikasi lisogenik)[8].



Gambar 1 Contoh *Clinical picture*
(Sumber: Cortes, dkk., 2008)

Setiap sel yang terinfeksi akan berkembang dengan cara melakukan replikasi litik atau replikasi lisogenik. Kedua replikasi tersebut dilakukan berdasarkan

probabilitas litik (p_{lt}) dan probabilitas lisogenik (p_{lg}) dengan total keduanya bernilai 1, jadi $p_{lt}+ p_{lg}= 1$. Pada mulanya, banyaknya *nucleus capsid* (NR untuk replikasi litik dan IT untuk replikasi lisogenik) bernilai nol (NR=0 dan IT=0).

A. Replikasi Lisogenik

Pada replikasi lisogenik, aktifasi proses mutasi berhenti setelah iterasi mencapai batas maksimal (LIT). Nilai LIT tergantung pada kondisi kesehatan sel, jadi sel yang sehat (fungsi minimasi (f(x)) yang bernilai tinggi) akan memiliki probabilitas infeksi rendah sehingga nilai LIT menjadi lebih tinggi. Sebaliknya, sel yang tidak sehat akan memiliki nilai LIT yang lebih rendah (Cortes, dkk., 2011). Nilai LIT dapat dihitung dari nilai LIT inisialisasi (LIT^0) berdasarkan nilai fungsi tujuannya relatif terhadap fungsi tujuan terbaik yang telah ditemukan, yaitu dapat dihitung dengan persamaan berikut.

$$\boldsymbol{LIT_{sel-x} = LIT^0 \left(\frac{f(x)-f(\hat{x})}{f(\hat{x})}\right)} \quad (1)$$

keterangan:
$LIT_{sel\text{-}x}$ = nilai LIT sel
LIT^0 = nilai inisial LIT
$f(x)$ = nilai fungsi tujuan pada sel x
$f(\hat{x})$ = nilai fungsi tujuan terbaik yang sudah diperoleh

B. Replikasi Litik

Pada replikasi litik, pertama perlu didapatkan banyaknya replikasi *nicleus capsid* (NR). NR dihitung untuk tiap iterasi sebagai fungsi dari variabel binomial (Z), nilainya ditambahkan pada NR yang telah ada pada *clinical picture*. Z dihitung dengan menggunakan distribusi binomial yang diberikan oleh nilai maksimum replikasi *nucleus capsid* (LNR) dan probabilitas pada satu replikasi (p_r).

$$z = \text{bin}(LNR, p_r) \quad (2)$$

$$NR_{iter} = NR_{iter-1}+z \quad (3)$$

LNR menggambarkan batas dinding sel pecah dan virus yang menghuninya keluar. Nilai LNR tergantung pada nilai fungsi tujuan yang diminimalkan (f(x)). Sel dengan nilai f(x) yang tinggi memiliki probabilitas terinfeksi lebih rendah, maka nilai LNR akan menjadi lebih tinggi [8]. Nilai LNR dapat dihitung dari nilai LNR inisialisasi (LNR^0) berdasarkan nilai fungsi tujuannya relatif terhadap fungsi tujuan terbaik yang telah ditemukan, yaitu dapat dihitung dengan persamaan beikut.

$$LNR_{sel-x} = LNR^0 \left(\frac{f(x)-f(\hat{x})}{f(\hat{x})}\right) \quad (4)$$

keterangan:
$LNR_{sel\text{-}x}$ = nilai LNR sel
LNR^0 = nilai inisial LNR
$f(x)$ = nilai fungsi tujuan pada sel x
$f(\hat{x})$ = nilai fungsi tujuan terbaik yang sudah diperoleh

Setelah itu, sel yang memiliki nilai NR melebihi LNR akan pecah sehingga virus terbebaskan. Setiap virus yang terbebaskan akan memiliki probabilitas (p_i) untuk menginfeksi sel-sel baru lainnya yang berada di sekitar. Jika kardinalitas lingkungan dari x didefinisikan sebagai |V(x)|, jumlah sel yang terinfeksi oleh virus di lingkungan dapat diperoleh dengan menggunakan random binomial dari nilai |V(x)| dan probabilitas p_i.

$$y= \text{bin}(|V(x)|, p_i) \quad (5)$$

Disisi lain, untuk memepertahankan dirinya dari pertumbuhan infeksi virus, organisme (gabungan sel) merespon dengan mengeluarkan antigen. Dalam *clinical picture*, tiap-tiap sel yang terinfeksi mengeluarkan antibodi dengan distribusi probabilitas bernoulli $A(x)=\text{Ber}(p_{an})$, dimana p_{an} merupakan probabilitas antibodi yang dikeluarkan oleh sel *x* dalam *clinical-picture*. Oleh karena itu, jumlah populasi sel yang terinfeksi mengeluarkan antibodi didasarkan dengan karakteristik distribusi binomial dengan parameter ukuran *clinical picture* (*n*) dan probabilitas mengeluarkan antibodi (p_{an}) [8].

$$A(\text{populasi}) = \text{Bin}(n,p_{an}) \quad (6)$$

C. Mutasi

Replikasi lisogenik memungkinkan algoritma untuk melakukan pencarian dengan cara mutasi. Mutasi memiliki tujuan untuk memunculkan solusi baru yang berbeda sama sekali dengan solusi sebelumnya agar dapat keluar dari lokal optimum [9].

Dari suatu solusi [4-6-2-3-5-1-7] dapat dimutasi menjadi solusi yang lain. Ada beberapa cara mutasi, di antaranya adalah sebagai berikut [9]:

1. *Flip* (membalik)

Jika dilakukan proses membalik terhadap segemen di antara 2 garis tegak rute mula-mula 4-6-|2-3-5-1|-7 menjadi 4-6-|1-5-3-2|-7.

2. *Swap* (menukar)

Dengan proses menukar pada titik ke-3 dan 6, rute mula-mula 4-6-|2-3-5-1|-7 akan menjadi 4-6-|1-3-5-2|-7.

3. *Slide* (menggeser)

Cara ini dilakukan dengan melakukan penggeseran rute di atas menjadi 4-6-|5-2-3-1|-7.

D. Neighborhood

Adapun replikasi litik memungkinkan algoritma untuk melakukan pencarian dengan cara *neighborhood*. *Neighborhood* ini merupakan metode penelusuran solusi yang berdekatan dengan suatu solusi bagus yang sudah ditemukan sebelumnya. Jadi metode ini menjadikan algoritma semakin cepat dalam pencarian lokal. Sel tetangga dapat didefinisikan dengan bertukar sepasang gen dalam genom sel sebelumnya. Mulai dari gen (paling kiri) pertama, dipertukarkan dengan gen kedua yang dihasilkan sel tetangga pertama. Sel tetangga kedua diperoleh dengan bertukar gen kedua dengan yang ketiga. Untuk sel yang genom terdiri dari gen n, prosedur ini diulang sampai (n-1) sel tetangga yang diperoleh [10].

Sebagai contoh solusi fisibel [1-2-3-4-5] memiliki solusi *neighborhood* sebagai berikut:

Tabel 1Contoh *neighborhood*

Solusi mula-mula	1-2-3-4-5
Job pertama ditukar dengan ke-2	2-1-3-4-5
Job ke-2 ditukar dengan ke-3	1-3-2-4-5
Job ke-3 ditukar dengan ke-4	1-2-4-3-5
Job ke-4 ditukar dengan ke-5	1-2-3-5-4

III. PENJELASAN SINGLE MACHINE TOTAL WEIGHTED TARDINESS PROBLEM

Single machine total weighted tardiness problem (SMTWTP) terkenal sebagai masalah perencanaan operasi.

Dalam SMTWTP, satu set pekerjaan j = (j_1, ..., j_n) perlu diproses pada mesin tunggal. Setiap job j_i terdiri dari operasi tunggal saja, yang melibatkan waktu proses $p_i>0$ ∀ j = 1,..., n. Pentingnya relativitas pekerjaan yang diungkapkan dengan bobot yang bernilai nonnegatif, $w_i>0$ ∀ j = 1, ..., n. Mesin melakukan proses hanya mungkin satu pekerjaan pada satu waktu, tidak ada proses pekerjaan paralel. Setiap pekerjaan j seharusnya selesai sebelum batas waktu d_i. Jika pekerjaan selesai melebihi waktu d_i, maka terjadi keterlambatan (*lateness*) dan dihitung sebagai fungsi berikut:

$$T_j = \max (s_j + p_j - d_j, 0) \quad (7)$$

Permasalahan ini juga dapat diformulasikan pada model *Integer Programming* sebagai berikut [11]:

$$\text{Minimasi TWT} = \sum_{j=1}^{n} \left(w_j . \max(0, s_j + p_j - d_j) \right) \quad (8)$$

$$\text{Subject to:} \quad s_j - s_{j'} >= p_j - MZ_{j,j'} \quad (9)$$

keterangan
s_i = waktu mulai dari tujuan pekerjaan j
p_i = waktu proses untuk pekerjaaan j
d_i = batas waktu selesainya pekerjaan j
T_i = keterlambatan pekerjaan j
w_j = bobot keterlambatan pekerjaan j

Dengan nilai M merupakan nilai besar positif dan $Z_{j,j'}$ merupakan variabel biner yang bernilai 1 jika pekerjaan j diikuti pekerjaan j', kalau tidak maka nilainya 0.

IV. PENERAPAN ALGORITMA

Berikut adalah *pseudocode*penerapan algoritma *Viral Systems* pada SMTWTP tersebut.

Procedure VS_SMTWTP(N_{max}, CPS, p_{lt}, p_i, p_{an}, *LNR*, *LIT*,*data*)
CP=∅
iterasi = 0
for*i* = 1 to CPS
/* Dapatkan solusi layak secara acak beserta tipe replikasinya
CP(*i*) = Random permutasi ()
CP(*i*).Tipe_replikasi = Dapatkan_Tipe_Replikasi_Acak (p_{lt})
F(*i*) = *SMTWTP* (*CP(i), data*)
Next
Fbest = min (F)
Do
iterasi = *iterasi*+1
For*c* =1 to *clinical_size*
If Tipe_Replikasi(*CP* (c)) = 'Lytic' **Then**
Litik (*c* , *CP*, p_{lt} ,p_i, p_{an}, p_r, *LNR*)
Else
Lisogenik (*c, CP, plt, LIT*)
End If
Fbest = min (F)
Next
LoopUntil*iterations* = *Nmax* or cek_gap(*CP*) = True
End VS_SMTWTP

procedure Replikasi_Litik (*c* , *CP*, p_{lt} ,p_i, p_{an}, p_r, *LNR*)
CS = *CP(c)* /* Solusi_Sebelumnya
/* Dapatkan banyaknya replikasi *nucleus capsid*
z = Dapatkan_Probabilitas_Random_Binomial (*LNR*, *pr*)
CP(*c*).*NR* = *CP*(*c*).*NR* + z
CS.LNR=LNR(F(c)-Fbest)/Fbest
/* Cek infeksi
if*CS.NR >CS.LNR* **then**
/* Dapatkan daftar VS dari solusi tetangga secara *descend* berdasarkan kesehatan sel
Vs = Neighbourhood (CS)
VAS = Dapatkan_Neighbourhood_Urut(*Vs*)
/* Dapatkan CP secara *ascend* berdasarkan kesehatan sel
CPA = Dapatkan_Clinical_Picture_Urut (*CP*)
i = 1
foreach*S'*∈*VAS*
if*i* <= |*CPA*| **and** i <= |VAS| **then**
a = nilai_random
b = nilai_random
if*a* < p_i**and***b* > p_{an}**then**
/* Ganti *CPA*(*i*) dengan solusi baru *CS'*
CPA(*i*) = *CS'*
CPA(*i*).Tipe_Replikasi =Dapatkan_Tipe_Replikasi_Random(p_{lt})
FA(*i*) = SMTWTP(*CPA*(*i*), *data*)
replace = true
end-if
i = *i* + 1
end-for
end-if
end Replikasi_Litik

procedure Replikasi_Lisogenik(*c, CP, plt, LIT*)
CS = *CP(c)*
CS.IT = *CS.IT* + 1
CS.LIT=LIT(F(c)-Fbest)/Fbest
if*CS.IT >CS.LIT'* **then**
s = Dapatkan_random_gen ()
/* Lakukan mutasi *CS*
CS_{NEW} = Mutasi(*CS*, *s*)
CS_{NEW}.Tipe_Replikasi = Dapatkan_Tipe_Replikasi (p_{lt})
F(*c*) = SMTWTP (CS_{NEW}, *data*)
End-if
end Replikasi_Lisogenik

V. HASIL EKSPERIMEN

Eksperimen dilakukan terhadap parameter dan data pengujian. Pengujian eksperimen dilakukan untuk mengetahui performansi parameter. Sedangkan pengujian pada data pengujian dilakukan untuk mengetahui performansi algoritma secara keseluruhan.

A. Eksperimen parameter

Eksperimen terhadap parameter dilakukan dengan melakukan skenario terhadap perubahan paramter. Berikut adalah skenario yang diterapkan untuk parameter-parameter tersebut.

Tabel 2Skenario parameter

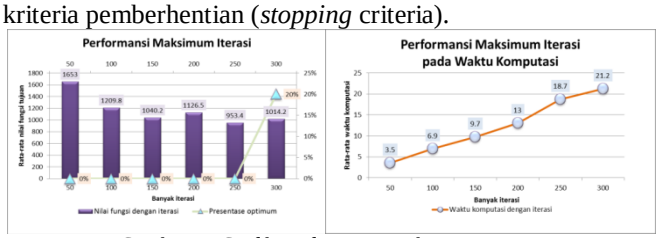
Para-meter	Skenario					
	ke-1	ke-2	ke-3	ke-4	ke-5	ke-6
Nmax	200	400	600	800	1000	1200
CPS	10	20	30	40	50	60

Para-meter	Skenario					
	ke-1	ke-2	ke-3	ke-4	ke-5	ke-6
Plt	0	0.2	0.4	0.6	0.8	1
Pi	0	0.2	0.4	0.6	0.8	1
Pan	0	0.2	0.4	0.6	0.8	1
Pr	0.2	0.4	0.6	0.8	1	-
LNR	5	10	15	20	25	30
LIT	5	10	15	20	25	30
Jenis Mutasi	swap	flip	fslide	bslide	acak	kom-binasi

Parameter dasar yang digunakan untuk mengiringi skenario parameter yang diuji tersebut adalah untuk Nmax = 200, CPS = 20, plt = 0.6, pi = 0.7, pan = 0.3, pr = 0.7, LNR = 5, danLIT = 5. Setiap parameter diujikan dengan repetisi10 kali agar dapat dianalisa performansi dan hasil yang didapatkan dengan beberapa skenario parameter. Selanjutnya dilakukan perbandingan antara beberapa parameter tersebut.

1. Maksimum iterasi (Nmax)

Nilai maksimum iterasi menunjukkan banyaknya perputaran algoritma. Nilai ini juga digunakan sebagai kriteria pemberhentian (*stopping* criteria).

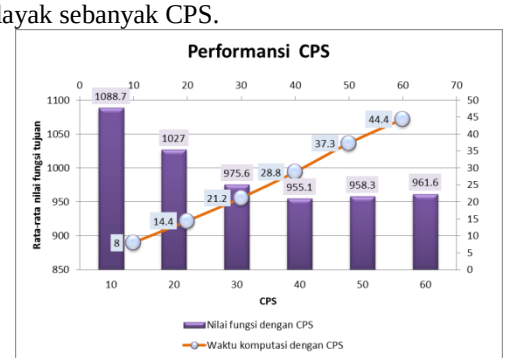


Gambar 2 Grafik performansi maksimum iterasi

Berdasarkan Gambar 2, terlihat bahwa semakin banyak iterasi, maka potensi untuk dapat menggapai global optimum menjadi semakin besar. Namun perubahan banyaknya iterasi pada nilai yang sudah tinggi tidak memberikan perubahan yang signifikan.Dalam hal waktu komputasi, semakin banyak nilai maksimum iterasi, maka waktunya akan semakin lama.

2. Ukuran Clinical Picture (CPS)

CPS menunjukkan ukuran banyaknya solusi yang dimuat dalam CP. Pada tahap inisialisasi, CP diperoleh secara acak solusi layak sebanyak CPS.

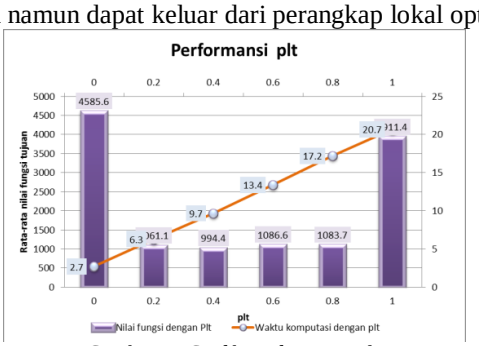


Gambar 3 Grafik performansi CPS

Berdasarkan hasil eksperimen tersebut, terlihat hasil pada Gambar 3bahwa semakin besari nilai CPS, menjadikan pencarian semakin baik dalam menggapai nilai optimum. Namun penambahan pada nilai yang sudah besar tidak memberikan perubahan hasil yang sifnifikan. Dalam hal komputasi, CPS dengan nilai yang besar menjadikan proses pencarian menjadi lebih lama.

3. Probabilitas replikasi litik (plt)

Probabilitas replikasi litik (plt) menunjukkan besarnya proporsi replikasi secara litik dan lisogenik yang beranalogi pada *neighborhood* dan mutasi. Jika nilai plt lebih dari 0,5 maka pencarian lebih banyak dilakukan secara *neighborhood*. Hal ini memicu diperolehnya nilai optimum, namun mungkin bersifat lokal. Jika nilai plt kurang dari 0,5 maka pencarian lebih banyak dilakukan dengan mutasi. Hal ini menjadikan pencarian lebih lambat menggapai nilai optimum namun dapat keluar dari perangkap lokal optimum.



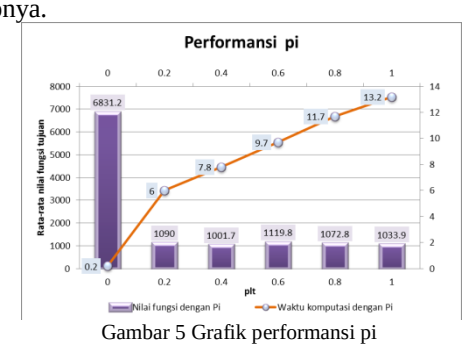
Gambar 4 Grafik performansi plt

Berdasarkan grafik hasil eksperimen yang ditunjukkan pada Gambar 4 ersebut, terlihat bahwa pada pengujian set data pertama pada 40 pekerjaan, probabilitas yang baik terletak sekitar angka 0,4. Nilai plt = 0 menunjukkan bahwa algoritma secara keseluruhan menggunakan metode mutasi. Sebaliknya nilai plt=1 menunjukkan bahwa algoritma secara keseluruhan menggunakan metode *neighborhood*. Penggunakan plt = 0 atau plt = 1 memberikan hasil yang buruk.

Oleh karena itu proporsi yang baik dalam plt menjadikan pencarian menjadi cepat dan dapat menggapai global optimum. Berdasarkan eksperimen ini juga dapat ditarik kesimpulan bahwa penggunaan dua jenis pencarian (*neighborhood* dan mutasi) merupakan kombinasi yang bagus.

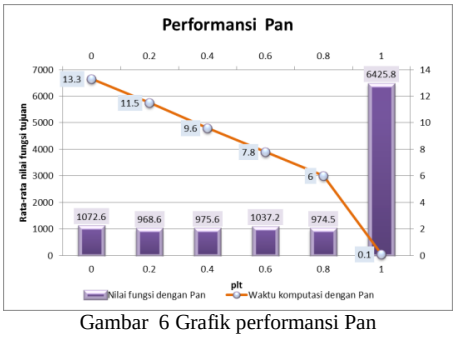
4. Probabilitas infeksi (pi) dan antigen (pan)

Probabilitas infeksi beranalogikan pada besarnya probabilitas virus melakukan infeksi terhadap sel yang berada disekitar sel yang sebelumnya dihinggapi. Sedangkan probabilitas antigen beranalogikan pada probabilitas sel mengeluarkan antibodi setelah virus melakukan infeksi terhadapnya.



Gambar 5 Grafik performansi pi

Dengan nilai yang pan yang sama (pan = 5), pada pengujian yang telah dilakukan nilai pi antara 0,2 hingga 1, tidak terlalu memberikan pengaruh signifikan. Namun apabila nilai pi =0, hasil yang didapatkan sangat buruk. Nilai pi = 0 menunjukkan analogi bahwa tidak ada satu virus yang menginfeksi sel tetangga, sehingga tidak terjadi *neighborhood*. Dalam hal waktu komputasi, semakin besar nilai pi maka waktu komputasi menjadi semakin lama.



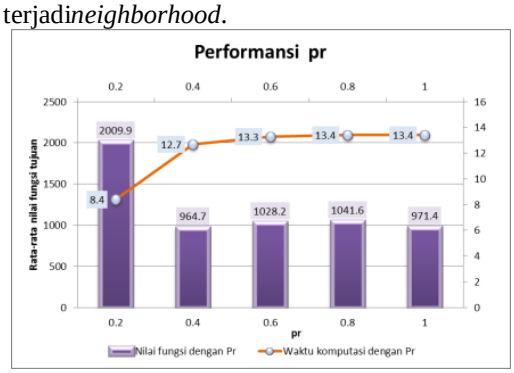
Gambar 6 Grafik performansi Pan

Dengan nilai yang pi yang sama (pi = 5), pada pengujian yang telah dilakukan nilai pan antara 0 hingga 0,8, tidak terlalu memberikan pengaruh signifikan. Namun apabila nilai pan = 1, hasil yang didapatkan sangat buruk. Nilai pan = 0 menunjukkan analogi bahwa semua sel tetangga mampu mencegah terjadinya infeksi, sehingga *neighborhood*-pun tidak terjadi. Dalam hal waktu komputasi, semakin besar nilai pi maka waktu komputasi menjadi semakin cepat.

Besarnya pi dan pan berpengaruh terhadap besarnya *neighborhood* terhadap solusi-solusi sekitar solusi yang sebelumnya telah pecah. Semakin besar pi, maka semakin besar kemungkinan *neighborhood*. Namun semakin besar pan, maka kemungkinan *neighborhood* menjadi semakin kecil.

5. Probabilitas replikasi tunggal (pr)

Probabilitas replikasi tunggal didasarkan pada analogi probabilitas *nucleus capsid* yang ada didalam sel melakukan replikasi. Semakin besar nilai pr, maka banyaknya *nucleus capsid* akan semakin banyak berkembang. Hal ini memicu virus akan cepat memecahkan sel sehingga terjadi *neighborhood*. Pada setiap iterasi, nilai NR semakin bertambah sebesar nilai random yang diperoleh dari pr dan besarnya LNR. Jadi pr ini berpengaruh terhadap seberapa sering terjadi*neighborhood*.

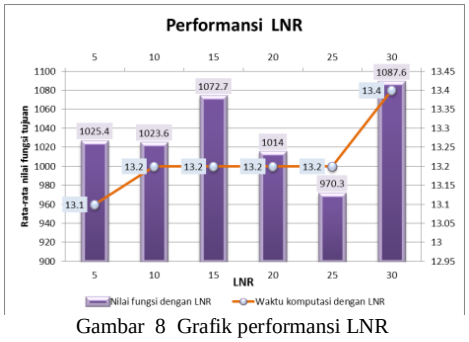


Gambar 7 Grafik performansi pr

Hasil eksperimen sebagaimana yang ditampilan pada grafik dalam Gambar 7, menunjukkan bahwa nilai pr yang terlalu kecil menyebabkan hasil yang didapatkan tidak dapat menggapai solusi yang baik. Hal ini dikarenakan terlalu jarangnya dilakukan *neighborhood*. Namun keuntungannya adalah waktu komputasi yang cepat.

6. LNR

Nilai LNR merupakan batas nilai NR. Jika nilai NR sudah melebihi LNR maka, *nucleus capsid* yang berada dalam sel akan memecahkan dinding sel dan keluar untuk melakukan infeksi terhadap sel yang berada disekitar. Hal ini dianalogikan pada solusi baru yang mendekati pada solusi sebelumnya. Nilai LNR suatu sel juga dianalogikan pada kesehatan sel. Sel dikatakan sehat jika memiliki nilai fungsi tujuan yang semakin buruk.



Gambar 8 Grafik performansi LNR

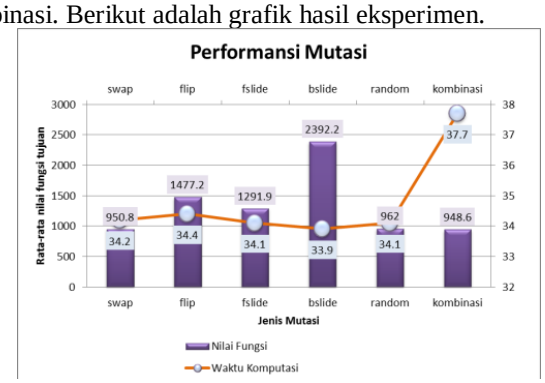
Pada Algoritma ini, nilai LNR sel didapatkan dari nilai fungsinya relatif terhadap nilai fungsi terbaik dikalikan pada LNR dasar. Parameter LNR pada dasarnya tidak berpengaruh signifikan terhadap algoritma, karena perkembangan nilai NR dipengaruhi oleh besarnya pr yang juga didasarkan pada nilai LNR.

7. LIT

Nilai LIT menunjukkan batas nilai IT. Jika nilai IT melebihi nilai LIT, maka *nucleus capsid* yang berada dalam sel dapat termutasi. Hal ini dianalogikan pada terjadinya mutasi dari solusi sebelumnya menjadi solusi baru. Nilai IT bertambah satu setiap kali iterasi, sedangkan nilai LIT diperoleh dari besarnya nilai fungsi sel relatif terhadap besarnya nilai fungsi terbaik dikalikan dengan LIT dasar. Parameter LIT ini menentukan besarnya frekuensi terjadinya mutasi.

8. Mutasi

Pada kasus kombinatorial, mutasi dapat dilakukan dengan beberapa cara yang di antaranya adalah *swap*, *flip*, dan *slide*. Pada panelitian ini juga dilakukan pengujian terhadap penggunaan mutasi dengan 6 skenario penggunaan mutasi, yaitu *swap*, *flip*, *forward slide*, *backward slide*, acak dan kombinasi. Berikut adalah grafik hasil eksperimen.



Gambar 9 Grafik performansi jenis mutasi

Grafik hasil eksperimen pada Gambar 9 tersebut, menunjukkan bahwa mutasi dengan cara *swap* pada kasus SMTWTP memberikan solusi yang paling baik. Penggunaan kombinasi jenis mutasi juga memberikan hasil yang bagus tapi membutuhkan waktu yang relatif lebih lama.

B. Eksperimen pada data pengujian

Algoritma *Viral Sistems* diterapkan untuk menyelesaikan set permasalahan 125 set data 40 pekerjaan, 125 set data 50 pekerjaan dan 25 set data 100 pekerjaan.

Berdasarkan eksperimen yang dilakukan untuk set data 40 pekerjaan, *Viral System* dapat menemukan 125 nilai optimum dari 125 data. Sebanyak 123 set data di antaranya ditemukan dengan menggunakan parameter maksimum iterasi 500, adapun 2 set data, yaitu 40_58 dan 40_85 ditemukan solusi optimumnya dengan

menambah nilai parameter maksimum iterasi menjadi 1000. Waktu komputasi yang diperlukan rata-rata 23.7 detik.

Eksperimen terhadap 50 pekerjaan, dihasilkan solusi optimum sebanyak 101 set data dari 125 set data 50 pekerjaan. Rata-rata lama waktu komputasi yang diperlukan adalah 41,6 detik. Adapun eksperimen terhadap set data 100 pekerjaan, Algoritma *Viral System* dapat menemukan solusi optimum sebanyak 9 dari 25 set data. Lama waktu komputasinyanya rata-rata 344 detik.

VI. KESIMPULAN

Berdasarkan hasil penelitian ini, maka dapat disimpulkan bahwa metode Algoritma *Viral System* dapat diterapkan untuk menyelesaikan *Single Machine Total Weighted Total Tardiness Problem* (SMTWTP).Hasil eksperimen komputasi menunjukkan bahwa *Viral Systems* baik digunakan untuk menyelesaikan permasalahan untuk 40 pekerjaan, tapi semakin meningkat banyaknya pekerjaan performansinya semakin menurun. Algoritma *Viral systems*inimemiliki 8 parameter masing-masingberpengaruh terhadap proses pencarian. Nilai maksimum iterasi bila semakin besar maka hasil yang didapatkan relatif semakin baik dan waktu yang diperlukan semakin lama. CPS jika semakin besar maka waktu komputasi semakin lama. Nilai plt yang semakin besar menjadikan algoritma lebih banyak melakuan *neighborhood* daripada mutasi. Parameter pi berkebalikan dengan parameter pan, semakin besar pi maka semakin besar peluang terjadi *neighborhood*. Pr menentukan tingkat seringnya dilakukan *neighborhood*. Besarnya nilai LNR tidak terlalu berpengaruh signifikan terhadap algoritma. Adapun parameter LIT, semakin besar nilainya akan semakin jarang dilakukan mutasi.

DAFTAR PUSTAKA

[1] M. L. Pinedo, *Scheduling Theory, Algorithms, and Systems*. New York: Springer, 2008.

[2] H. Nazif, "A Genetic Algorithm on Single Machine Scheduling Problem to Minimise Total Weighted Completion Time," *European Journal of Scientific Research*, vol. 35, pp. 444-452, 2009.

[3] J. K. Lenstra, A. H. G. R. Kan, and P. Brucker, "Complexity of machine scheduling problems," *Annals of Discrete Mathematics*, vol. 1 pp. 343–362, 1997.

[4] C. N. Potts and L. N. V. Wassenhove., "A branch and bound algorithm for the total weighted tardiness problem," *Operations Research*, vol. 33 pp. 363–377, 1985.

[5] R. K. Congram, C. N. Potts, and S. L. v. d. Velde, "An Iterated Dynasearch Algorithm for the Single-Machine Total Weighted Tardiness Scheduling Problem," *INFORMS J. on Computing*, vol. 14, pp. 52-67, 2002.

[6] X. Wang and L. Tang, "A Population-Based Variable Neighborhood Search for The Single Machine Total Weighted Tardiness Problem," *Computers and Operations Research*, vol. 36, pp. 2105-2110, 2009.

[7] P. Cortés, J. M. García, J. Muñuzuri, and L. Onieva, "Viral systems : A new bio-inspired optimisation approach," *Computers & Operations Research*, vol. 35, pp. 2840 - 2860, 2008.

[8] P. Cortes, J. M. Garcia, J. Munuzuri, and J. Guadix, "A viral system massive infection algorithm to solve the Steiner tree problem in graphs with medium terminal density," *Industrial Engineering*, vol. 2, pp. 71-77, 2010.

[9] B. Santosa and P. Willy, *Metode Metaheuristik, Konsep dan Implementasi*. Surabaya: Guna Widya, 2011.

[10] D. Suryadi and E. K. Kartika, "Viral Systems Application for Knapsack Problem," presented at the Proceedings of the 2011 Third International Conference on Computational Intelligence, Communication Systems and Networks, 2011.

[11] B. Santosa and M. A. Budiman, "Simple Algorithm for Single Machine Total Weighted Tardiness Problem," in *Proceeding of Industrial Engineering and Service Science, 2011, September 20-21*, Solo, 2011.